Project 6: Randomization and Matching

Stacy Chen and Sofia Guo

```
# Load tidyverse and MatchIt
# Feel free to load other libraries as you wish
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(MatchIt)
library(ggplot2)
library(cobalt)
```

```
##  cobalt (Version 4.5.4, Build Date: 2024-02-26)
##
## Attaching package: 'cobalt'
##
## The following object is masked from 'package:MatchIt':
##
##     lalonde
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(optmatch)
library(dplyr)
# Load ypsps data
ypsps <- read_csv('/Users/stacyworkuser/Downloads/ypsps.csv')
```

```
## Rows: 1254 Columns: 174
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (174): interviewid, college, student_vote, student_meeting, student_othe...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(ypsps)
```

```
## # A tibble: 6 x 174
##   interviewid college student_vote student_meeting student_other student_button
##         <dbl>   <dbl>        <dbl>           <dbl>         <dbl>          <dbl>
## 1           1       1            1               0             0              0
## 2           2       1            1               1             1              1
## 3           3       1            1               0             0              1
## 4           4       0            0               0             0              0
## 5           5       1            1               1             0              0
## 6           6       1            1               0             0              0
## # i 168 more variables: student_money <dbl>, student_communicate <dbl>,
## #   student_demonstrate <dbl>, student_community <dbl>, student_ppnscal <dbl>,
## #   student_PubAff <dbl>, student_Newspaper <dbl>, student_Radio <dbl>,
## #   student_TV <dbl>, student_Magazine <dbl>, student_FamTalk <dbl>,
## #   student_FrTalk <dbl>, student_AdultTalk <dbl>, student_PID <dbl>,
## #   student_SPID <dbl>, student_GovtOpinion <dbl>, student_GovtCrook <dbl>,
## #   student_GovtWaste <dbl>, student_TrGovt <dbl>, student_GovtSmart <dbl>, ...
```

```
options(scipen = 999)
```

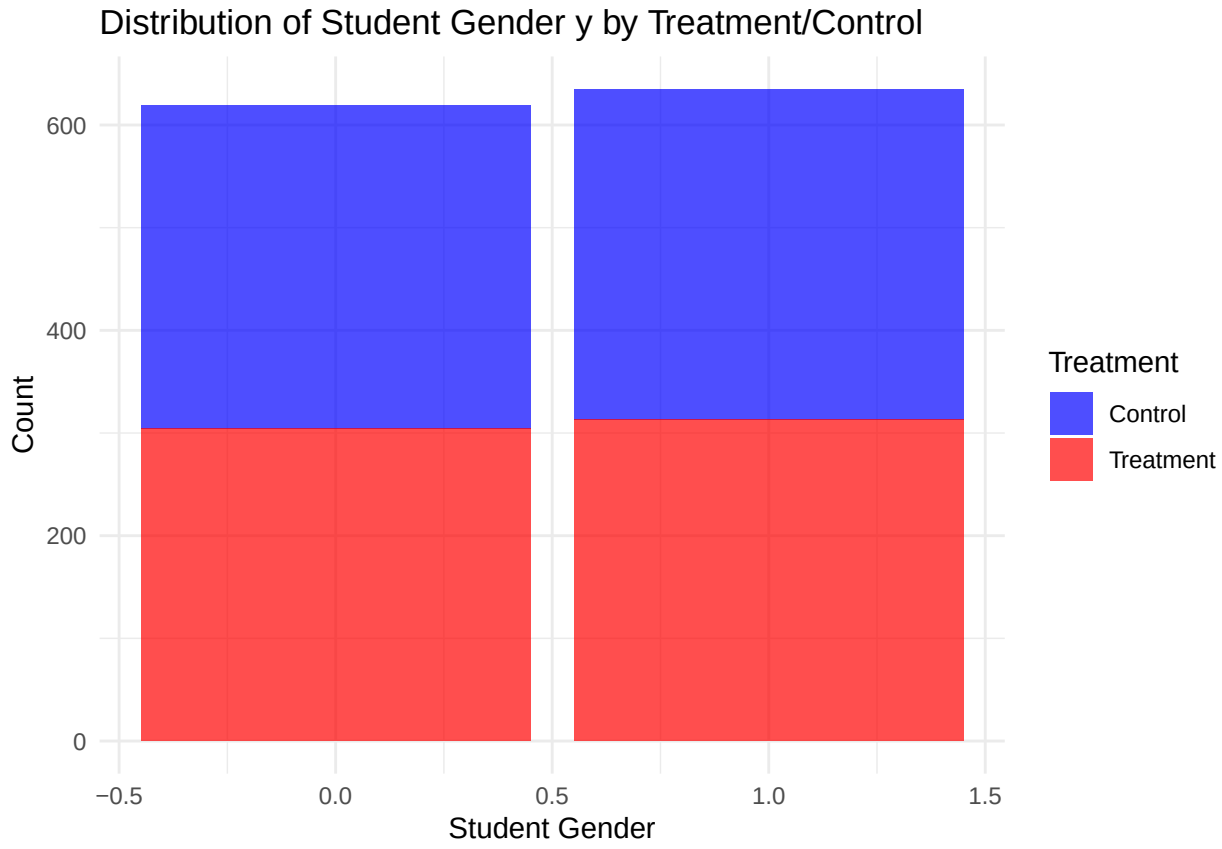
Randomization

```
# Generate a vector that randomly assigns each unit to treatment/control
set.seed(123)
ypsps <- ypsps %>%
  mutate(treatment = as.numeric(rbernoulli(length(unique(interviewid)), p = 0.5)))
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `treatment = as.numeric(rbernoulli(length(unique(interviewid)),
##   p = 0.5))`.
## Caused by warning:
## ! `rbernoulli()` was deprecated in purrr 1.0.0.
```

```
baseline_cov <- ypsps %>%
    select(student_Gen, treatment)
# Choose a baseline covariate (use dplyr for this)
baseline_cov <- ypsps %>%
    select(student_Gen, treatment)
# Visualize the distribution by treatment/control (ggplot)
ggplot(baseline_cov, aes(x = student_Gen, fill = factor(treatment))) +
```

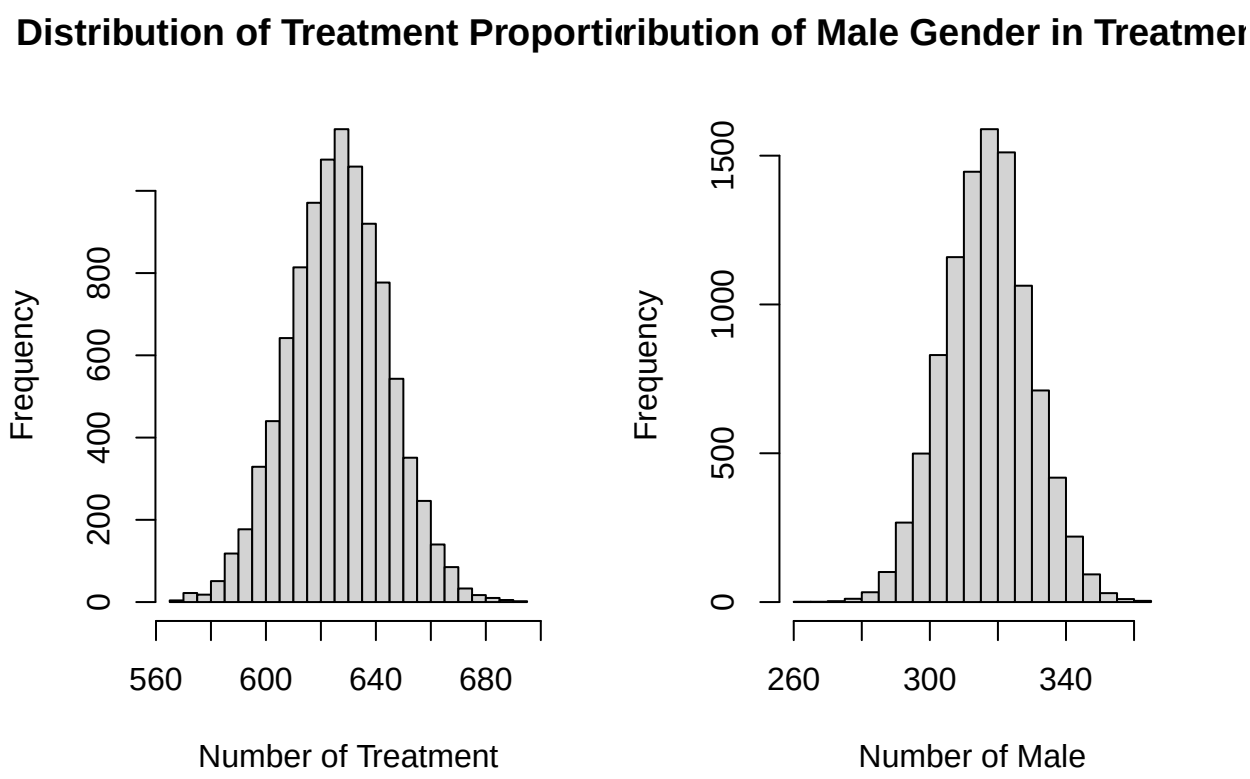
```
  geom_bar(position = "stack", alpha = 0.7) +
  labs(x = "Student Gender", y = "Count", fill = "Treatment", title = "Distribution of Student Gender y
  scale_fill_manual(values = c("blue", "red"), labels = c("Control", "Treatment")) +
  theme_minimal()
```



Simulation

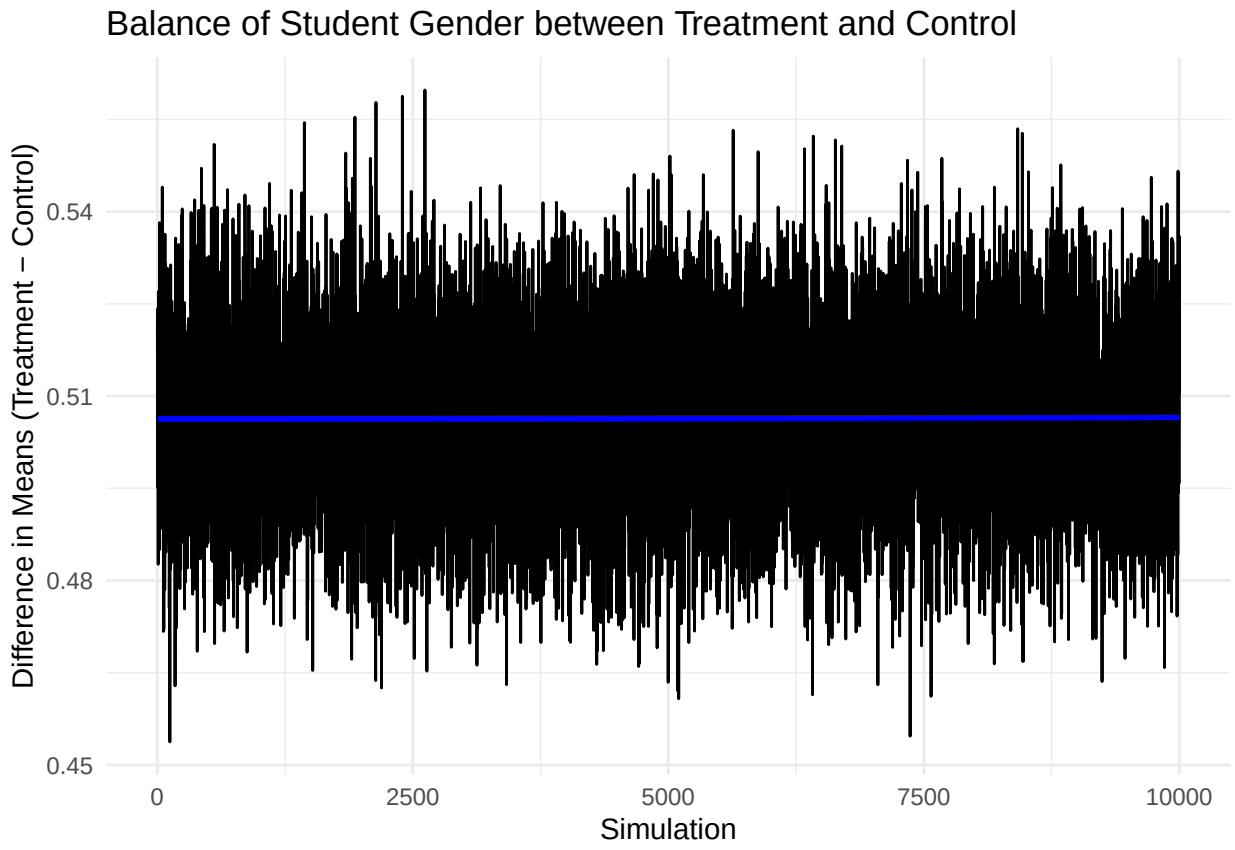
```
# Simulate this 10,000 times (monte carlo simulation - see R Refresher for a hint)
n_simulations <- 10000

sim_results <- matrix(nrow = n_simulations, ncol = 2)

# Perform Monte Carlo simulation
for (i in 1:n_simulations) {
  # Generate treatment assignment vector
  df <- ypsps %>%
    select(interviewid,student_Gen) %>%
    mutate(treatment = as.numeric(rbernoulli(length(unique(interviewid)), p=0.5)))

  # Calculate the proportion of treatment units
  proportion_treatment <- sum(df$treatment)

  # Calculate the proportion of Male gender
  proportion_male <- sum(df$student_Gen[df$treatment == 1])

  # Store the results
```

```
  sim_results[i, 1] <- proportion_treatment
  sim_results[i, 2] <- proportion_male
}

par(mfrow = c(1, 2)) # Arrange plots in one row and two columns
hist(sim_results[, 1], breaks = 30, main = "Distribution of Treatment Proportions",
     xlab = "Number of Treatment", ylab = "Frequency")
hist(sim_results[, 2], breaks = 30, main = "Distribution of Male Gender in Treatment Group",
     xlab = "Number of Male", ylab = "Frequency")
```




```
sim_results_data <- as.data.frame(sim_results)
names(sim_results_data) <- c("proportion_treatment", "proportion_male")

# Calculate the difference in means (Treatment - Control) for each simulation
sim_results_data$diff_means <- ifelse(sim_results_data$proportion_treatment == 0,
                                      0,
                                      sim_results_data$proportion_male / sim_results_data$proportion_tr

# Plot the difference in means against simulation index
ggplot(sim_results_data, aes(x = 1:n_simulations, y = diff_means)) +
  geom_line() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add a linear trend line
  labs(x = "Simulation", y = "Difference in Means (Treatment - Control)", title = "Balance of Student G
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Questions

1. **What do you see across your simulations? Why does independence of treatment assignment and baseline covariates not guarantee balance of treatment. assignment and baseline covariates?**

I see a roughly normal distribution in my simulation in both number of treatment and number of males (gender=1). The balance of Student Gender between Treatment and Control with 10,000 simulations is 50/50. Independence of treatment assignment and baseline covariates doesn't guarantee balance if there is only one draw because of random chance. If the draw happens many times, we can expect it to converge to balance.

Propensity Score Matching

One Model: using gender, race, gpa, student plan for attending school, parent education, parent income, parent newspaper reading

```
# Select covariates that represent the "true" model for selection, fit model
ypsps_covariates <- ypsps %>%
```

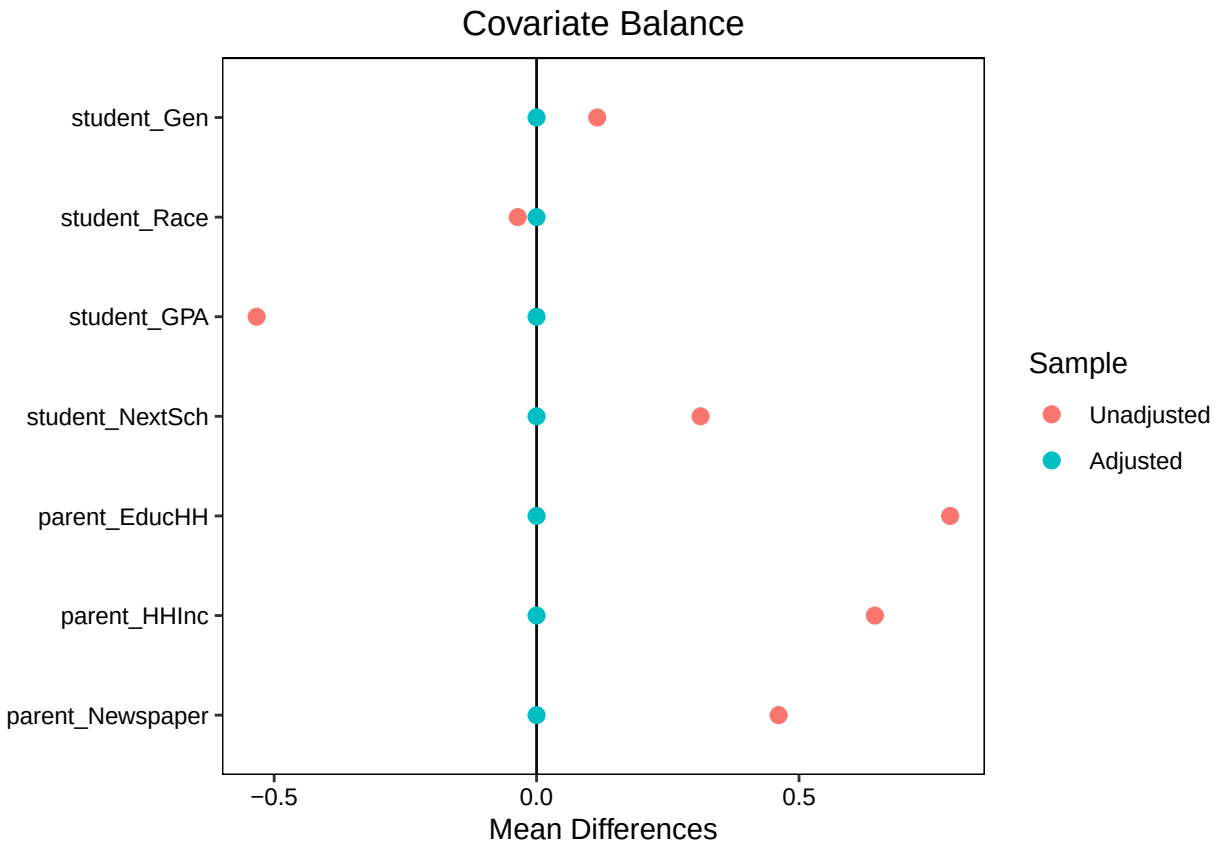
```
    select(interviewid, college, student_Gen, student_Race, student_GPA, student_NextSch, parent_EducHH

glm_model <- glm(formula = college ~ student_Gen + student_Race + student_GPA + student_NextSch + paren

ypsps_covariates$propensity_score <- predict(glm_model, type = "response")

match_exact_att <- matchit(formula= college ~ student_Gen + student_Race + student_GPA + student_NextSc

# Report the overall balance and the proportion of covariates that meet the balance threshold
match_summ <- summary(match_exact_att, un=F)

# Filter covariates based on SMD threshold
balanced_covariates <- match_summ$sum.matched[abs(match_summ$sum.matched[, "Std. Mean Diff."]) < 0.1, ]

# Print balanced covariates
print(balanced_covariates)
```

```
##                   Means Treated Means Control           Std. Mean Diff.
## student_Gen           0.5722892     0.5722892 -0.00000000000000002220446
## student_Race          1.0120482     1.0120482 -0.00000000000000002220446
## student_GPA           2.4487952     2.4487952 -0.00000000000000017763568
## student_NextSch       0.9608434     0.9608434 -0.00000000000000002220446
## parent_EducHH         2.9216867     2.9216867 -0.00000000000000022204460
## parent_HHInc          7.2409639     7.2409639 -0.00000000000000044408921
## parent_Newspaper      3.7228916     3.7228916 -0.00000000000000022204460
##                   Var. Ratio               eCDF Mean                eCDF Max
## student_Gen               NA 0.00000000000000002220446 0.00000000000000002220446
## student_Race       0.9899347 0.00000000000000010049831 0.00000000000000010087417
## student_GPA        0.9899347 0.00000000000000007147061 0.00000000000000010035375
## student_NextSch           NA 0.00000000000000002220446 0.00000000000000002220446
## parent_EducHH      0.9899347 0.00000000000000005457151 0.00000000000000007719519
## parent_HHInc       0.9899347 0.00000000000000002910866 0.00000000000000007632783
## parent_Newspaper   0.9899347 0.00000000000000002092077 0.00000000000000007615436
##                   Std. Pair Dist.
## student_Gen                     0
## student_Race                    0
## student_GPA                     0
## student_NextSch                 0
## parent_EducHH                   0
## parent_HHInc                    0
## parent_Newspaper                0
```

Covariate plot

```
#covariate plot
match_exact_att <- matchit(formula= college ~ student_Gen + student_Race + student_GPA + student_NextSc

#make covariate plot
love.plot(match_exact_att)
```

```
## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the 'stars' argument to distinguish between them and appropriately label the x-axis.
```



All variables met the threshold, it looks balanced on the plot, I will keep all my covariates. Now, calculate ATT:

```
#estimate the ATT using linear regression
match_exact_att_data <- match.data(match_exact_att)

# model
lm_full_att <- lm(student_ppnscal ~ college + student_Gen + student_Race + student_GPA + student_NextSc

#summarize results
lm_full_att_summ <- summary(lm_full_att)

#calculate ATT
ATT_full <- lm_full_att_summ$coefficients["college","Estimate"]
ATT_full
```

```
## [1] 0.9228414
```

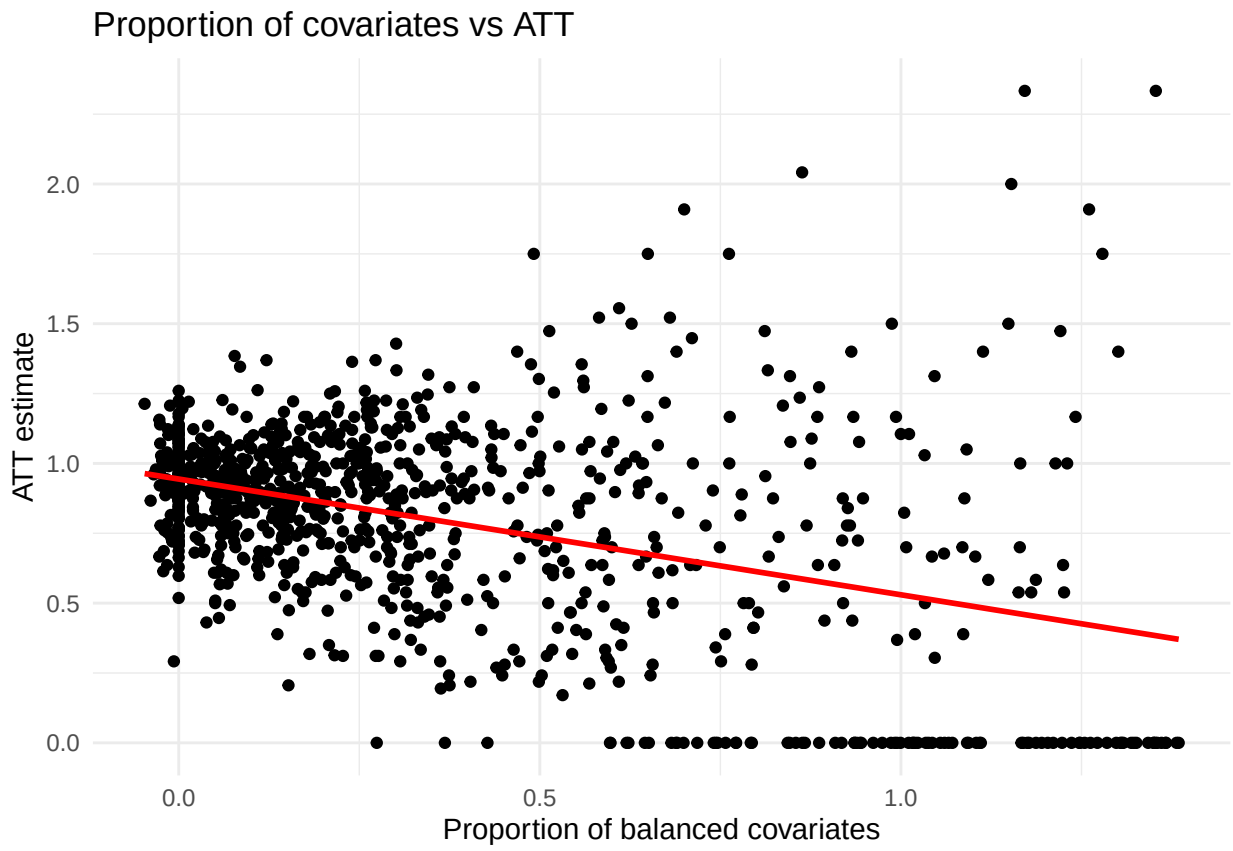
Simulations with random covariates

```
# Data manipulation: rename post treatment covariates with "post_", create list of post_vars names, and
df_renamed <- ypsps %>%
  rename_with(~paste0("post_", .), contains("1973") | contains("1983") | contains("1982"))
```

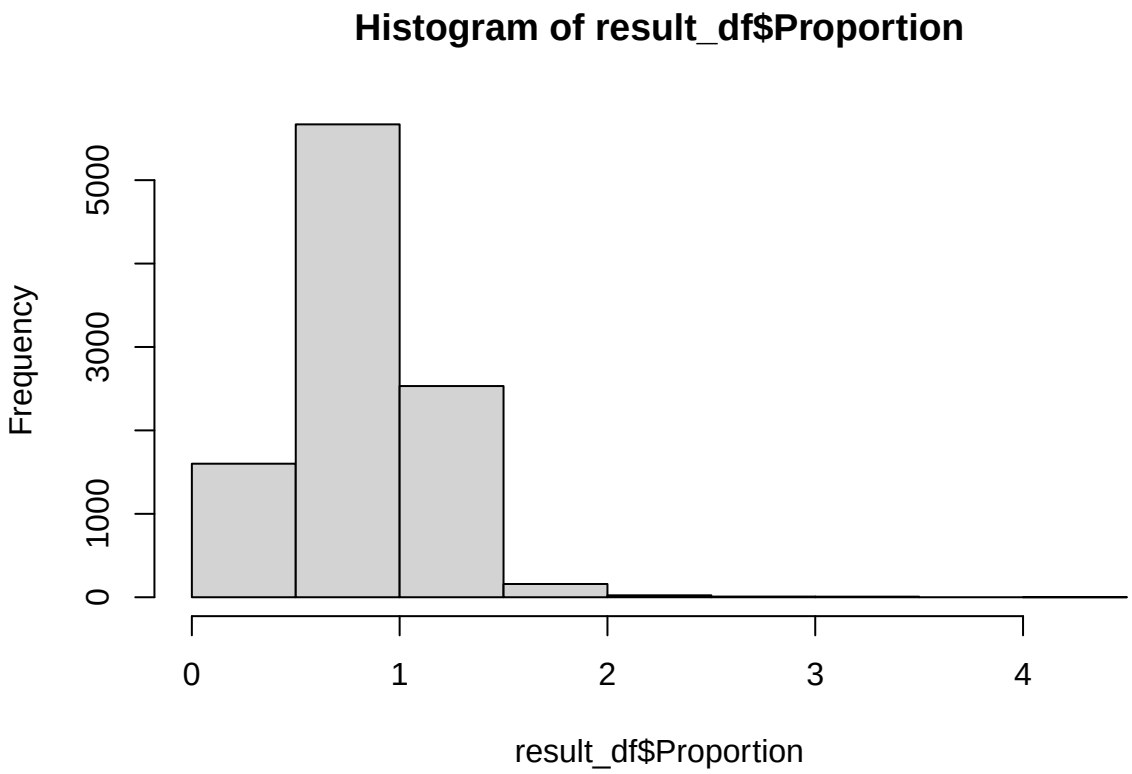
```r
# Get list of post-treatment variable names
post_vars <- names(df_renamed) %>%
  keep(~str_starts(., "post_"))

# Create prevars_df excluding post-treatment variables and specific placebo variables
prevars_df <- df_renamed %>%
  select(-any_of(c(post_vars, "interviewid", "treatment", "parent_GPHighSchoolPlacebo", "parent_HHColleg
  filter_all(any_vars(!is.na(.)))  # Filter out rows with any NA values

# Get names of pre-treatment variables
pre_vars <- colnames(prevars_df)

# Initialize an empty matrix to store results
result_matrix <- matrix(nrow = 10000, ncol = 3)
colnames(result_matrix) <- c("ATT", "Proportion", "Improvement")

# Simulate random selection of features 10k+ times
for (i in 1:10000) {
 suppressWarnings({ # Randomly select the number of covariates
  num_covariates <- sample(1:length(pre_vars), 1)
  # Randomly choose covariates
  random_covariates <- sample(pre_vars, num_covariates)
  # Select the columns
  df <- df_renamed %>%
    select(college, student_ppnscal, all_of(random_covariates)) %>%
    filter(complete.cases(.))
  # Fit the propensity score model (assuming glm for simplicity)
  model <- glm(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))), data = df)
  # Match treated and control units
  match_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))), data =
# Report the overall balance and the proportion of covariates that meet the balance threshold
    match_summ <- summary(match_att, un=F)
    # Filter covariates based on SMD threshold
    balanced_covariates <- match_summ$sum.matched[abs(match_summ$sum.matched[, "Std. Mean Diff."]) < 0.
  proportion_true <- length(balanced_covariates) / length(random_covariates)
  match_exact_att_data <- match.data(match_att)
  #define covariates
  covariates <- random_covariates
  matched_df <- match_exact_att_data
  smd_before <- sapply(df[, covariates], function(x) {
  (mean(x[df[["college"]] == 1],na.rm=T) - mean(x[df[["college"]] == 0],na.rm=T)) /
  sqrt((var(x[df[["college"]] == 1]) + var(x[df[["college"]] == 0])) / 2)
  })
  # Calculate SMD after matching
  smd_after <- sapply(df[, covariates], function(x) {
    (mean(x[matched_df[["college"]] == 1],na.rm=T) - mean(x[matched_df[["college"]] == 0],na.rm=T)) /
    sqrt((var(x[matched_df[["college"]] == 1]) + var(x[matched_df[["college"]] == 0])) / 2)
  })
  # Calculate mean percent improvement
  mean_percent_improvement <- mean((smd_before - smd_after) / smd_before * 100, na.rm = TRUE)
    # model
    model <- lm(as.formula(paste("student_ppnscal ~ college +", paste(random_covariates, collapse = "+")
 #summarize results
```

```
    lm_full_att_summ <- summary(model)
    #calculate ATT
    ATT <- lm_full_att_summ$coefficients["college","Estimate"]
 })
  result_matrix[i, ] <- c(ATT, proportion_true, mean_percent_improvement)

}

# Fit p-score models and save ATTs, proportion of balanced covariates, and mean percent balance improve

# Plot ATT v. proportion
result_df <- as.data.frame(result_matrix)
subsample_df <- result_df[sample(nrow(result_df), 1000), ]
ggplot(subsample_df, aes(ATT, Proportion)) +
  geom_point()+
    geom_smooth(method = "lm", se = FALSE, color = "red") +  # Add trend line without confidence interv
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Proportion of covariates vs ATT", x = "Proportion of balanced covariates", y = "ATT esti
  theme_minimal()
```

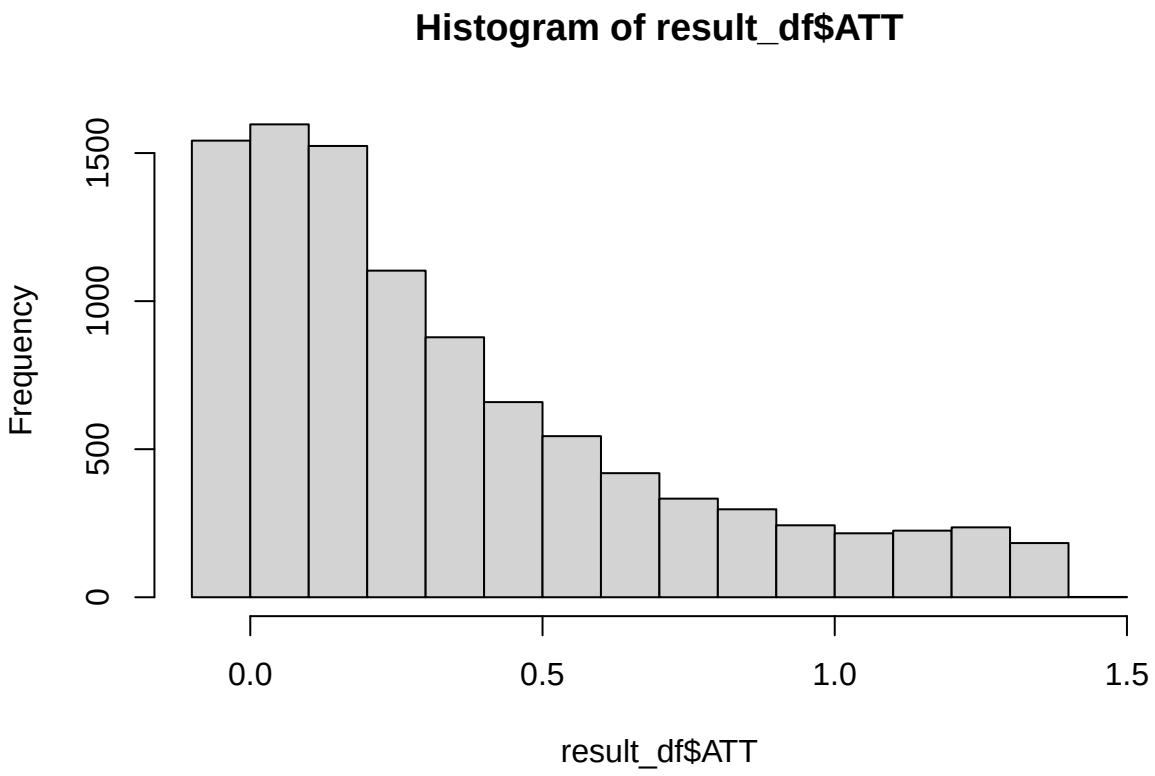
```
## `geom_smooth()` using formula = 'y ~ x'
```




```
hist(result_df$Proportion)
```




```r
hist(result_df$ATT)
```




```
#empty list of love plots
match_list <- list()

# Set up loop to iterate 10 times
for (i in 1:10) {
  # Randomly select the number of covariates
  num_covariates <- sample(1:length(pre_vars), 1)

  # Randomly choose covariates
  random_covariates <- sample(pre_vars, num_covariates)

  # Select the random columns
  df <- df_renamed %>%
    select(interviewid, college, student_ppnscal, all_of(random_covariates))

  # Step 3: Calculate ATT
  # Match treated and control units
  match_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))), data =

  # Store the results in the result matrix
match_list[[i]] <- love.plot(match_att)}
```

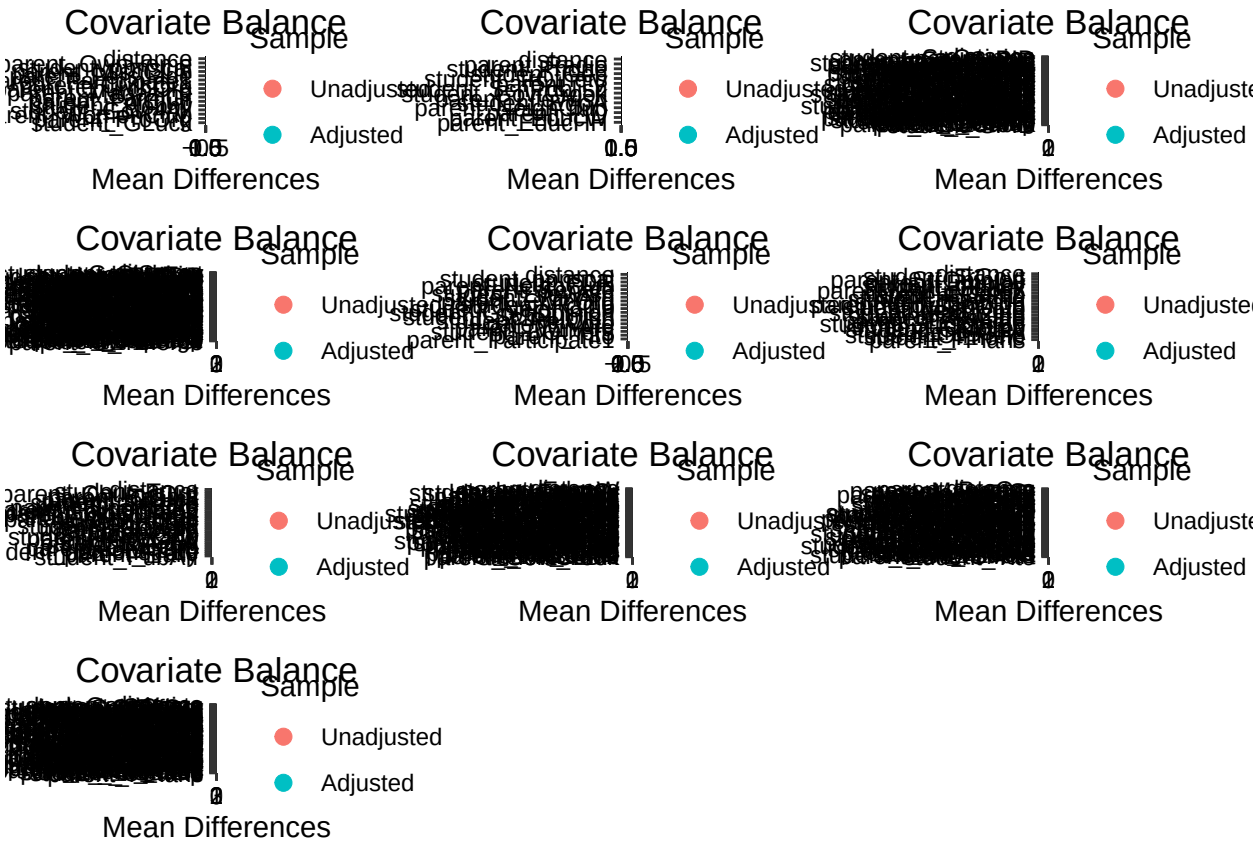
```
## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
```

```
## Use the ‘stars‘ argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the ‘stars‘ argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the ‘stars‘ argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the ‘stars‘ argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the ‘stars‘ argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the ‘stars‘ argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the ‘stars‘ argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the ‘stars‘ argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the ‘stars‘ argument to distinguish between them and appropriately label the x-axis.
```

```
## Warning: Fewer control units than treated units; not all treated units will get
## a match.
```

```
## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the 'stars' argument to distinguish between them and appropriately label the x-axis.
```

```
grid.arrange(grobs = match_list, ncol = 3)
```




```
#count number of simulations where balanced covariate proportion was higher
#find mean proportion
meanprop <- mean(result_df$Proportion)
#filter for higher than mean proportion
higherprop <- result_df %>%
  filter(Proportion > meanprop)
#count number of simulations
nrow(higherprop)
```

```
## [1] 5763
```

Questions

1. **How many simulations resulted in models with a higher proportion of balanced covariates? Do you have any concerns about this?** From looking at the histogram of proportions, 57

2. **Analyze the distribution of the ATTs. Do you have any concerns about this distribution?** Your Answer: ATT graph is right skewed, I believe that suggests that the true treatment effect is positive but underestimated since the models are not correctly specified. It also has a wide range, which shows me that a bad model can lead to wonky results down the line.

3. **Do your 10 randomly chosen covariate balance plots produce similar numbers on the same covariates? Is it a concern if they do not?** Your Answer: It's hard to tell whether there are similar numbers on the same covariates based on covariate balance plots alone since there are such a varying number of covariates. I can see that the covariate balance is very inconsistent so it would be a good way for me to know that my results are biased.

Matching Algorithm of Your Choice

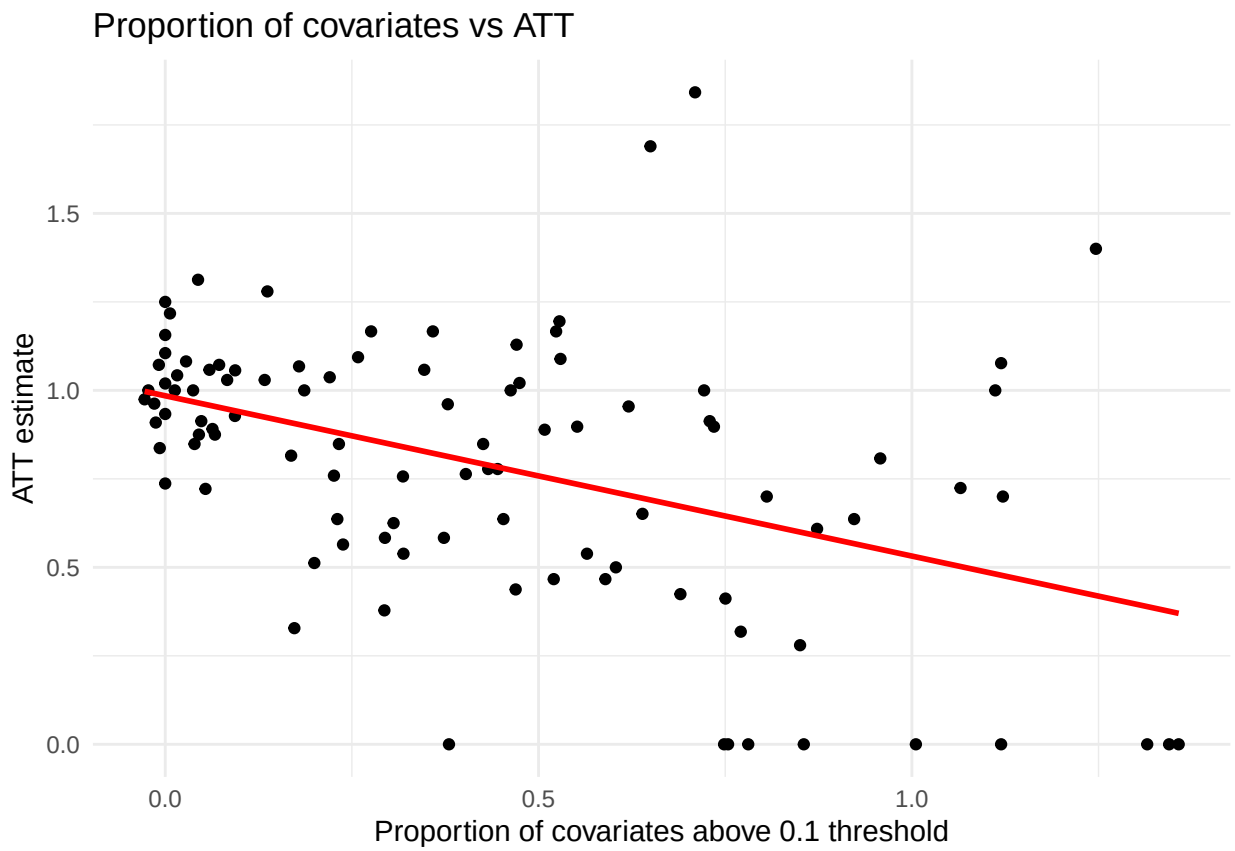
Simulate Alternative Model: Nearest Neighbor

```
# Initialize matrix
result_matrix_1 <- matrix(nrow = 10000, ncol = 3)
colnames(result_matrix_1) <- c("ATT", "Proportion", "Improvement")

# Simulate random selection of features 1000 times
for (i in 1:10000) {
  suppressWarnings({
    # Randomly select the number of covariates
    num_covariates <- sample(1:length(pre_vars), 1)
    # Randomly choose covariates
    random_covariates <- sample(pre_vars, num_covariates)
    # Select the columns
    df <- df_renamed %>%
      select(college, student_ppnscal, all_of(random_covariates)) %>%
      filter(complete.cases(.))
    # Fit the propensity score model using KNN matching
    match_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))),
                         data = df,
                         method = "nearest",
                         distance = "glm",
                         link = "logit",
                         discard = "control",
                         replace = FALSE,
                         ratio = 2)
    # Report the overall balance and the proportion of covariates that meet the balance threshold
    match_summ <- summary(match_att, un = FALSE)
    # Filter covariates based on SMD threshold
    balanced_covariates <- match_summ$sum.matched[abs(match_summ$sum.matched[, "Std. Mean Diff."]) < 0.
    proportion_true <- length(balanced_covariates) / length(random_covariates)
    match_exact_att_data <- match.data(match_att)
    # Define covariates
    covariates <- random_covariates
    matched_df <- match_exact_att_data
    smd_before <- sapply(df[, covariates], function(x) {
      (mean(x[df[["college"]] == 1], na.rm = TRUE) - mean(x[df[["college"]] == 0], na.rm = TRUE)) /
        sqrt((var(x[df[["college"]] == 1]) + var(x[df[["college"]] == 0])) / 2)
    })
```

```
    # Calculate SMD after matching
    smd_after <- sapply(df[, covariates], function(x) {
      (mean(x[matched_df[["college"]] == 1], na.rm = TRUE) - mean(x[matched_df[["college"]] == 0], na.rm
        sqrt((var(x[matched_df[["college"]] == 1]) + var(x[matched_df[["college"]] == 0])) / 2)
    })
    # Calculate mean percent improvement
    mean_percent_improvement <- mean((smd_before - smd_after) / smd_before * 100, na.rm = TRUE)
    # Fit linear model
    model <- lm(as.formula(paste("student_ppnscal ~ college +", paste(random_covariates, collapse = "+"
    # Summarize results
    lm_full_att_summ <- summary(model)
    # Calculate ATT
    ATT <- lm_full_att_summ$coefficients["college", "Estimate"]
  })
    # Store results in matrix
    result_matrix_1[i, ] <- c(ATT, proportion_true, mean_percent_improvement)
}
```

```
# Plot ATT v. proportion
result_df_1 <- as.data.frame(result_matrix_1)

subsample_df <- result_df_1[sample(nrow(result_df_1), 100), ]
ggplot(subsample_df, aes(ATT, Proportion)) +
  geom_point()+
    geom_smooth(method = "lm", se = FALSE, color = "red") +  # Add trend line without confidence interv
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Proportion of covariates vs ATT", x = "Proportion of covariates above 0.1 threshold", y
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```r
#empty list of love plots
match_list <- list()

# Set up loop to iterate 10 times
for (i in 1:10) {
  # Randomly select the number of covariates
  num_covariates <- sample(1:length(pre_vars), 1)

  # Randomly choose covariates
  random_covariates <- sample(pre_vars, num_covariates)

  # Select the random columns
  df <- df_renamed %>%
    select(interviewid, college, student_ppnscal, all_of(random_covariates))

  # Step 3: Calculate ATT
  # Match treated and control units
  match_att <- matchit(as.formula(paste("college ~", paste(random_covariates, collapse = "+"))),
                       data = df,
                       method = "nearest",
                       distance = "glm",
                       link = "logit",
                       discard = "control",
                       replace = FALSE,
                       ratio = 2)
```

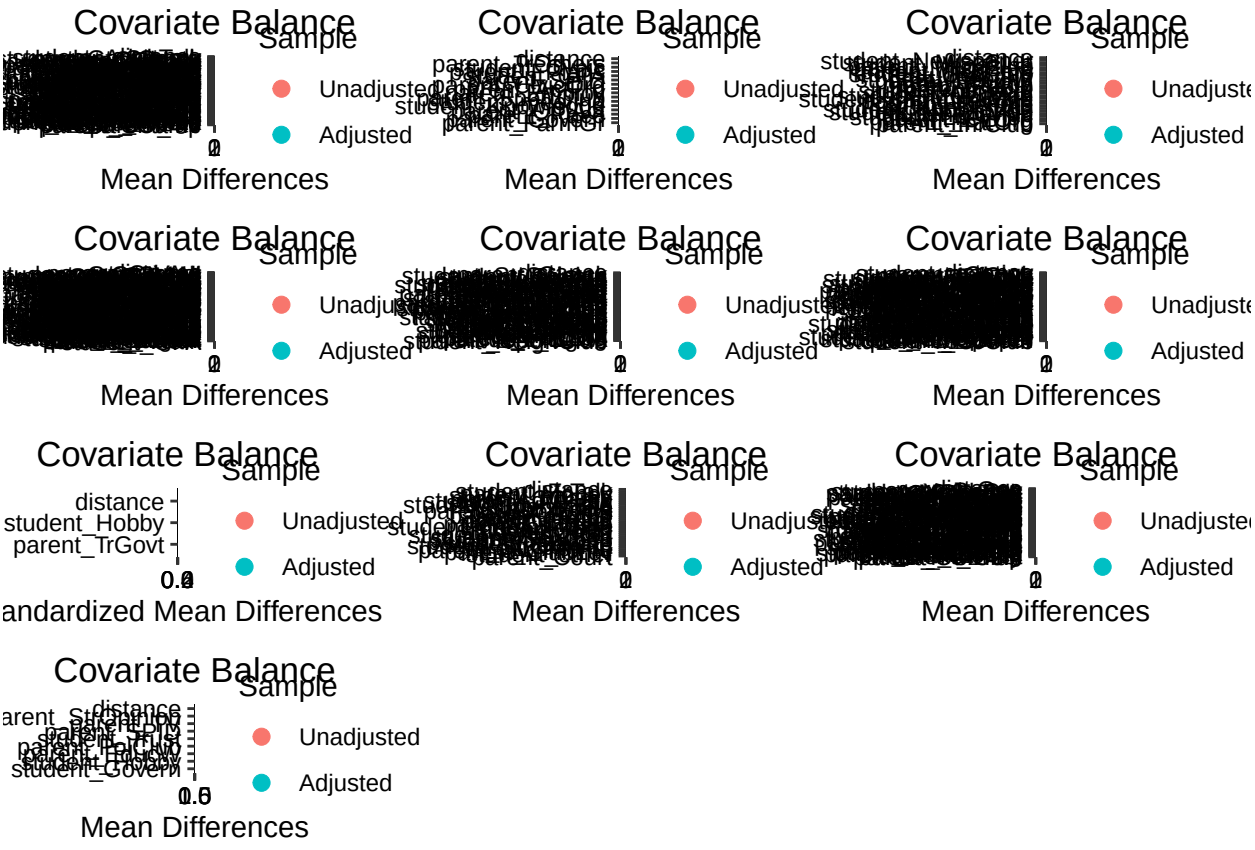
```r
  # Store the results in the result matrix
match_list[[i]] <- love.plot(match_att)}
```

```
## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the `stars` argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the `stars` argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the `stars` argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the `stars` argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the `stars` argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the `stars` argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get a match.
## Fewer control units than treated units; not all treated units will get a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the `stars` argument to distinguish between them and appropriately label the x-axis.

## Warning: Fewer control units than treated units; not all treated units will get
## a match.

## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the `stars` argument to distinguish between them and appropriately label the x-axis.
```

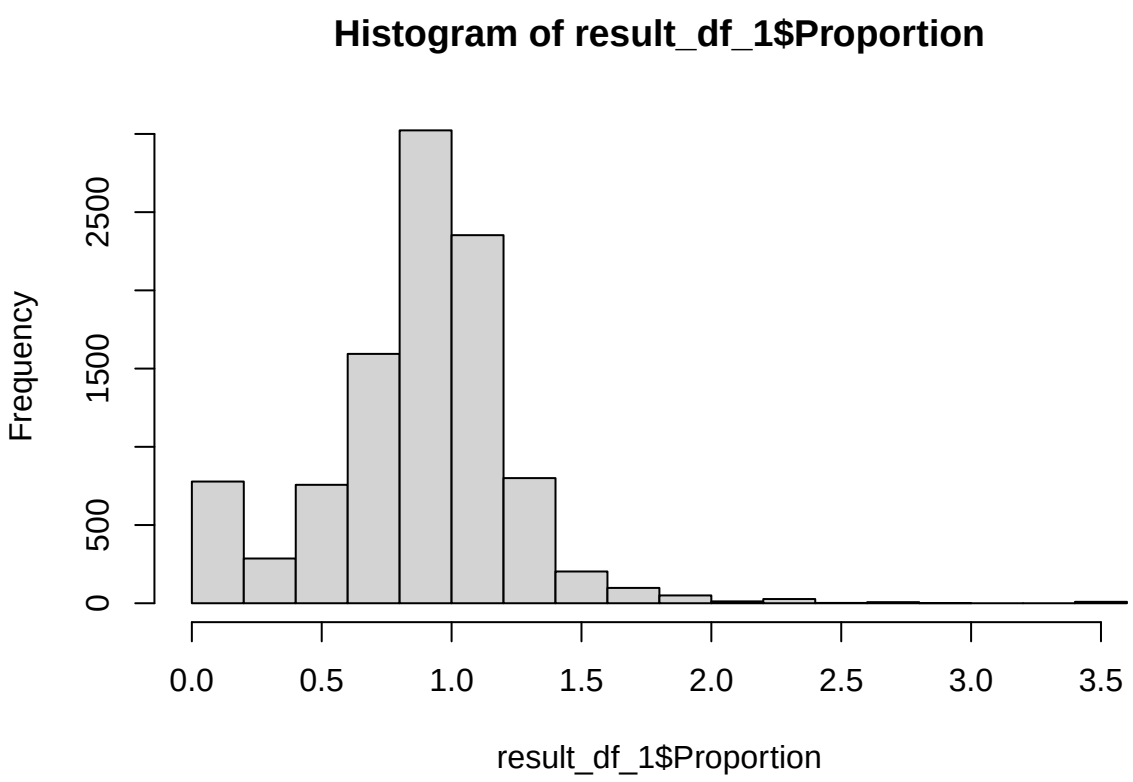
```
## Warning: Fewer control units than treated units; not all treated units will get
## a match.
```

```
## Warning: Standardized mean differences and raw mean differences are present in the same plot.
## Use the 'stars' argument to distinguish between them and appropriately label the x-axis.
```

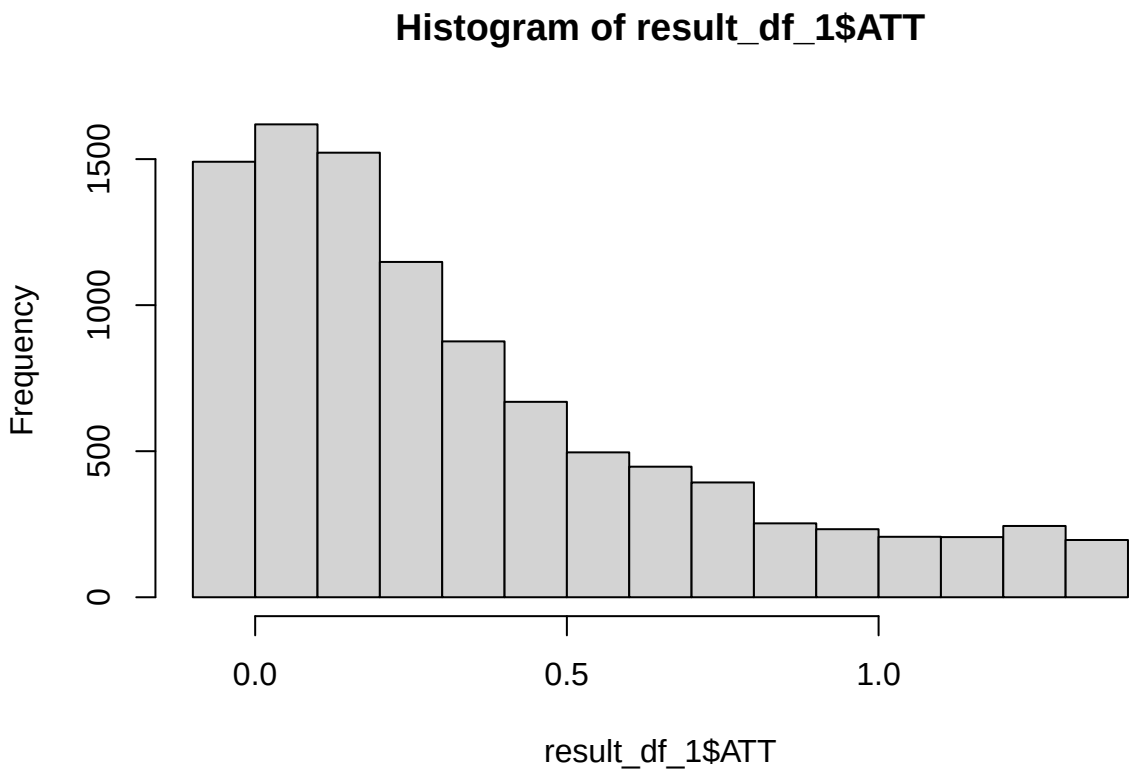
```
grid.arrange(grobs = match_list, ncol = 3)
```



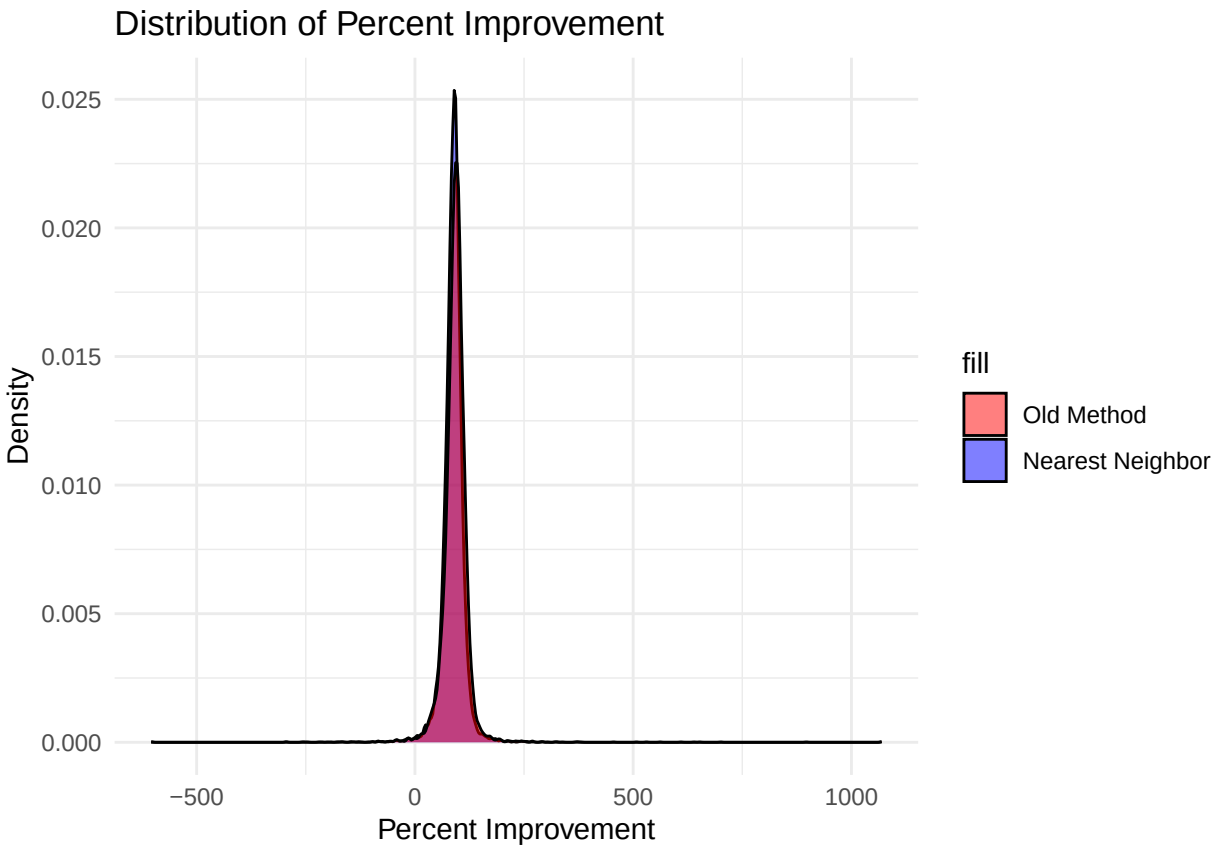

```
hist(result_df_1$Proportion)
```




```
hist(result_df_1$ATT)
```




```r
# Create density plots for the old method and the new method
old_method_plot <- ggplot(result_df, aes(x = Improvement)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(title = "Distribution of Percent Improvement (Old Method)", x = "Percent Improvement", y = "Dens
  theme_minimal()

new_method_plot <- ggplot(result_df_1, aes(x = Improvement)) +
  geom_density(fill = "red", alpha = 0.5) +
  labs(title = "Distribution of Percent Improvement (New Method)", x = "Percent Improvement", y = "Dens
  theme_minimal()

combined_plot <- ggplot() +
  geom_density(data = result_df, aes(x = Improvement, fill = "Old Method"), alpha = 0.5) +
  geom_density(data = result_df_1, aes(x = Improvement, fill = "Nearest Neighbor"), alpha = 0.5) +
  labs(title = "Distribution of Percent Improvement", x = "Percent Improvement", y = "Density") +
  scale_fill_manual(values = c("Old Method" = "blue", "Nearest Neighbor" = "red"), labels = c("Old Metho
  theme_minimal()

# Display the combined plot
print(combined_plot)
```




```
#count number of simulations where balanced covariate proportion was higher
#find mean proportion
meanprop <- mean(result_df_1$Proportion)
#filter for higher than mean proportion
higherprop <- result_df_1 %>%
  filter(!is.na(Proportion)) %>%  # Remove rows with NA in Proportion column
  filter(Proportion > meanprop)
# Count number of simulations
num_higherprop <- nrow(higherprop)
```

Questions

1. **Does your alternative matching method have more runs with higher proportions of balanced covariates?** Your Answer: I got 5811 nearest neighbor matching compared to 5744 for propensity scoring. The new method has higher proportions of balanced covariates (more above mean)

2. **Use a visualization to examine the change in the distribution of the percent improvement in balance in propensity score matching vs. the distribution of the percent improvement in balance in your new method. Which did better? Analyze the results in 1-2 sentences.** The distribution of the KNN method is not as spread out as propensity score matching method and is centered at the same mean. KNN is a better method since the spread is smaller. The percent improvement mean is the same though, meaning that it's likely that KNN can achieve better results with fewer simulations needed. To me, it's a more reliable matching method.

Discussion Questions

1. **Why might it be a good idea to do matching even if we have a randomized or as-if-random design?** Your Answer: There might be a bias where randomization doesn't evenly distribute baseline covariates evenly. It's a good way to conduct sensitivity analysis to ensure that a randomization process is providing unbiased results, since even when we have randomization, there might be unbalanced covariates to bias the treatment effects.

2. **The standard way of estimating the propensity score is using a logistic regression to estimate probability of treatment. Given what we know about the curse of dimensionality, do you think there might be advantages to using other machine learning algorithms (decision trees, bagging/boosting forests, ensembles, etc.) to estimate propensity scores instead?** Yes it might be better to use other ML algorithms to match because we can calibrate based on overfitting concerns. For example, BART could reduce the weight of covariates that are less important and provide better estimates that are more generalizable to new data.